BASIC Stamp programming course (4)

Part 4: Obstacle Alley

If you drive an automobile, you know the practical application of the Pauli exclusion principle: Two objects can't occupy the same space at the same time. What's true for automobiles is even truer for robots. An autonomous robot has to keep itself from colliding with obstacles. Obstacles might take the form of a wall or post, or they may be mobile like a dog, a person, or another robot.

By Al Williams

17

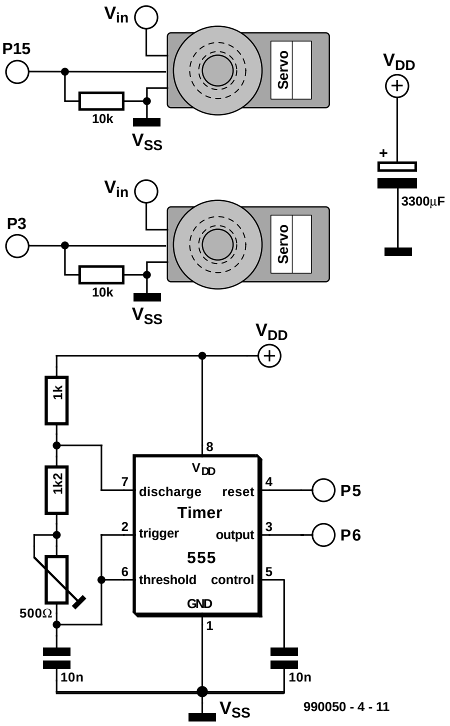

Figure 17. The oscillator and motor drive circuits.

Since the robot can't know the positions of moving objects in advance, it must have some way of detecting obstacles in real time. Humans, of course, use vision. While a robot that can see would be very desirable, it is also quite expensive and difficult to make a vision system appropriate for robotics.

Luckily, detecting obstacles doesn't require anything as sophisticated as machine vision. A much simpler system will suffice. Some robots use RADAR or SONAR (sometimes called SODAR when used in air instead of water). An even simpler system is to use infrared light to illuminate the robot's path and determine when the light reflects off an object. Thanks to the proliferation of infrared (IR) remote controls, IR illuminators and detectors are easily available and inexpensive.

In this instalment, I'll show you how to add infrared sensors to the Boe-Bot and use them to allow the robot to navigate around objects in its path. Along the way, you'll also see how to program some simple math into the BASIC Stamp and generate random numbers. You'll find the **parts list** close to circuit diagram Figure 18.

IR BASICS

In theory, detecting an object with IR is simple. You simply shine an IR light (an LED) in the forward direction and use a detector to look for the reflected light. In practice, it is somewhat more complicated. If you used this oversimplified approach, the detector will falsely trigger from ambient IR that occurs naturally.

To prevent these false triggers, you'll want to employ detectors that are sensitive to IR modulated at a particular frequency and modulate the IR source to that same frequency. In common remote controls for consumer electronics, the modulation frequency is 38 kHz. You can readily find IR receivers sensitive to this frequency.

While the BASIC Stamp could modulate the light source, the BASIC Stamp can't drive the LED and read the detector at the same time. Therefore, you'll want an external circuit to modulate the LED.

Another refinement useful for the robot is to use two detectors (and possibly two LEDs). You'll place one detector on the left side of the robot and the other on the right. This allows you to detect object and determine its position relative to the robot. If only one detector activates, the object is on that side. If both detectors turn on, the object is dead ahead.

THE OSCILLATOR

IR LEDs are commonplace and work just like regular LEDs. To modulate the LEDs, you can use a 555-based oscillator (see **Figure 17**). The LEDs will operate when input P5 (555 reset) is high. The oscillator output connects to the BASIC Stamp's pin 6. This is not necessary for operation, but it does allow you to calibrate the oscillator to operate at 38 kHz.

To perform the calibration, load the program in **Listing 4** and run it. This displays the frequency in Hertz. You can adjust the frequency by adjusting the potentiometer. The output should be near 38000. Of course, the 555 is not extremely stable, so the value may vary a bit, but it should be close to 38 kHz.

Since IR LEDs don't emit visible light, you might wonder if the circuit is working at all. You should be able to measure the LED's forward voltage across the LED (about 1.2 V). If you measure closer to 5 V, the LEDs are probably installed the wrong way around (usually, the long lead is the anode; the pin that connects to the BASIC Stamp).

Another way to troubleshoot IR LEDs is with a special plastic card that allows you to see the infrared light. Service technicians use these cards to test remote controls. You can buy them where you buy television repair supplies (you can also get them at Radio Shack/Tandy stores). For an electronic alternative, see 'Infra-red Remote Control Tester' in the July/August 1998 issue of *Elektor Electronics*.

PARTS LIST

Qty	Part
1	Operational Boe Bot
1	3300µF 16V electrolytic capacitor
1	500Ω potentiometer
2	0.01µF capacitor
2	0.1µF capacitor
1	1kΩ 1/8W or 1/4W resistor
1	1.2kΩ resistor
2	IR LEDs (LD271 or similar)
2	Infrared receiver (Panasonic 4602 or equivalent)
2	470Ω 1/4W resistor
1	555 Timer IC

IR DETECTION

Once you have the LEDs operational, you'll need to connect the IR detectors (see **Figure 18**). The detectors look like transistors with a bulge on one side. The bulge is the sensitive area. Although these devices have three terminals, they are not transistors. Two of the three pins carry power and ground to the detector. The other pin emits a logic 0 when it detects IR light. Here, a Panasonic type 4602 should be used.

The mounting of the detectors and the LEDs can be a little tricky and depends on the exact construction of your robot. You need to direct the LEDs in the forward direction and minimize the leakage from around the sides of the LEDs. You might consider using a bit of heat shrink sleeving, or cut small tubes from a soda straw, and cover it with electrical tape.

You'll also find that the position of the detectors is crucial. You should mount them as far apart as possible and tilt them slightly away from the LEDs (see **Figure 19**). To experiment with the positioning of the LEDs and detectors, use the program in **Listing 5**. When there is no object in front of the robot, you should not get an indication from the program. If you get false readings you probably have IR light leaking from the sides of the LEDs.

Once you have the detectors aligned properly, it is a simple matter to control the robot's motion based on the input from the detectors. **Listing 6** shows a simple control program that moves forward until it detects something. When the robot finds an obstacle, it stops and turns to avoid the object in its path.

MORE SOPHISTICATED

Once you have the simple control program under your belt, consider the program in **Listing 7**. This program is a bit more complex. Like the original program, this one moves forward until it detects an object. If the object is only visible on the right or left sensor, the robot turns in the opposite direction. If the object is straight ahead, the robot moves backwards and turns either right or left.

This program uses random numbers in several places. First, it uses a random number to select what direction to turn after it reverses in the face of an oncoming object. Also, any time the robot turns to the right or the left, it turns for a random amount of time.

18

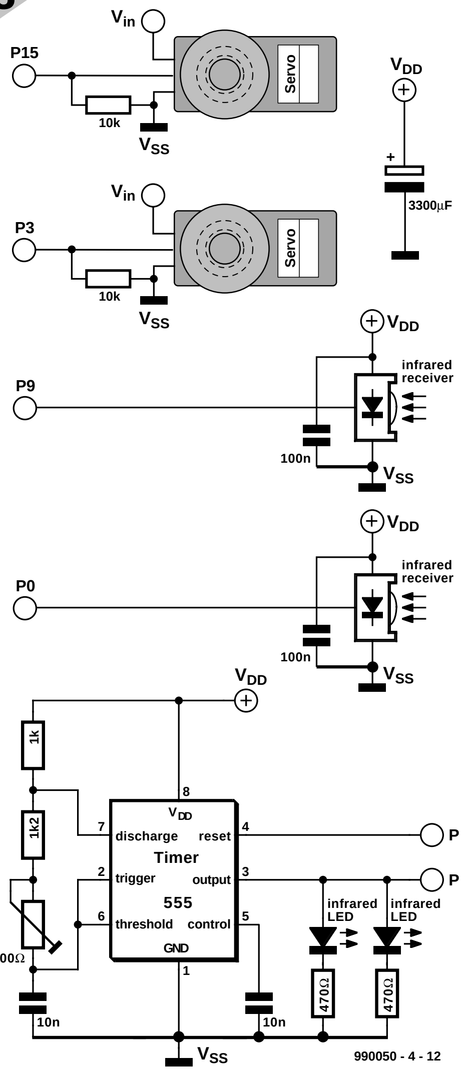

Figure 18. The complete circuit.

RANDOM NUMBERS

The BASIC Stamp uses the RANDOM command to generate pseudo random numbers. This command takes a number in a variable and manipulates it to produce another number. Although the number is random, you will get the same random number each time you start with the same number.

In this case, random means randomly distributed in the mathematical sense of the word. If you want an unpredictable number, you'll have to make the original number (the seed) vary unpredictably. The robot program adds 1 to the *rnd* variable each time through the main loop. That means that when the program calls *RANDOM*

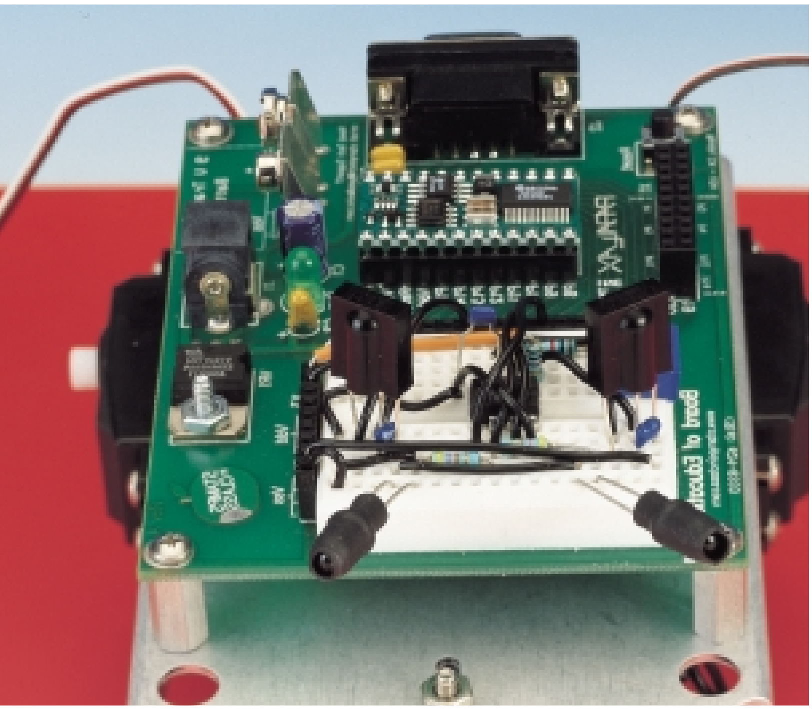

Figure 19. Suggested LED and sensor positioning.

Listing 4. Calibrating the 555 timer

```
freq var word
high 5  ‘ osc on
start:
  count 6,100,freq
  debug dec 5 freq*10,cr
  goto start
```

Listing 5. This Program aids in aligning the sensors

```
high 5              ‘osc on
top:
if in9=1 then noleft
debug ”left”
noleft:
debug 9  ‘ tab
if in0=1 then noright
debug ”right”
noright
debug cr
goto top
```

you won’t know what the seed value is and therefore won’t be able to predict the result.

Because the random number is distributed over the entire possible range of numbers, you should use a *WORD* variable with *RANDOM*. You can then use an AND (&) or mod (//) operator to limit the value of the number. For example, if you want to limit the number to 15 (%1111 in binary), you could write:

```
RANDOM rnd
X=rnd & %1111
```

This only works because 15 is one less than a power of 2. A more general-purpose way to limit the number is to use the mod operator (//). This operator returns the remainder from integer division. So 11//3 = 2 because 11/3 is 3 with a remainder of 2). To limit a random number to a maximum value of 100, you could write:

```
RANDOM rnd
X=rnd // 100
```

Armed with these simple equations, it is easy to generate random numbers for any range you need. Be careful not to restrict the *rnd* variable in the above example. Doing so will restrict the seed value for the next random number and therefore limit the range of future numbers the command will produce.

MATH EXPRESSIONS

Several spots in the robot program use expressions that have more than a single operator. These require special attention on the BASIC Stamp. When you write:

```
5+2*3
```

is the answer 21 or is the answer 11? In school, you learn that multiplication takes precedence over addition, so the answer is 11. However, the BASIC Stamp didn’t go to the same school you did, so it evaluates math expressions from left to right with no regard to common rules of precedence. So to the BASIC Stamp, the correct answer is 21.

Luckily, the BASIC Stamp does allow you to use parenthesis to group expressions together. To get the above expression to evaluate as a mathematician expects, you could rewrite it as:

```
5+(2*3)
```

or:

```
2*3+5
```

As an example, consider the portion of Listing 7 that causes the robot to turn left (just below the *left* label). Here, the

Listing 6. A Simple Motion Control Program

```
‘ Collision-avoiding robot - Williams
i var word        ‘ loop counter
right_IR var in0  ‘ right IR eye
left_IR var in9   ‘ left IR eye
right_servo con 3 ‘ right servo motor
left_servo con 15 ‘ left servo motor
IR_out con 5      ‘ 38kHz enable
delay con 10      ‘ motor cycle time
center con 750
speed con 100

high IR_out  ‘ turn on IR
pause 50

sense:
 if left_IR=0 or right_IR=0 then turn
‘ nothing in sight, so go forward
forward:

  for i=1 to delay*2
    pulsout left_servo,center-speed
    pulsout right_servo,center+speed
    pause 20
  next
  goto sense

turn:
  pause 50
  for i=1 to delay*5
    pulsout left_servo,center-speed
    pulsout right_servo,center-speed
    pause 20
  next
  goto sense
```

Listing 7. A Complex Motion Control Program

```
‘ Collision-avoiding robot - Williams
rnd var word        ‘ random quantity
i var word          ‘ loop counter
right_IR var in0    ‘ right IR eye
left_IR var in9     ‘ left IR eye
right_servo con 3   ‘ right servo motor
left_servo con 15   ‘ left servo motor
IR_out con 5        ‘ 38kHz enable
delay con 10        ‘ motor cycle time
center con 750
speed con 100

high IR_out         ‘ turn on IR
pause 50

sense:
  rnd=rnd+1         ‘ bump random number
‘ if obstacle in front of both eyes then
‘ go back and turn
 if left_IR=0 and right_IR=0 then back
‘ if obstacle to the left,
‘ move right or vice versa
 if left_IR=0 then right
 if right_IR=0 then left
‘ nothing in sight, so go forward
forward:

  for i=1 to delay*2
    pulsout left_servo,center-speed
    pulsout right_servo,center+speed
    pause 20
  next
  goto sense
back:
  pause 50
 for i=1 to delay*3
    pulsout left_servo,center+speed
    pulsout right_servo,center-speed
    pause 20
  next
‘ randomly turn to the right
‘ or the left after going
‘ backwards
  random rnd
  if rnd&1 then right


left:
  pause 50
  random rnd
  ‘ turn a random amount
  for i=1 to delay + (rnd&3*5)
    pulsout left_servo,center-speed
    pulsout right_servo,center-speed
    pause 20
  next
  goto sense

right:
  pause 50
  random rnd
‘ turn a random amount
 for i=1 to delay + (rnd&3*5)
    pulsout left_servo,center+speed
    pulsout right_servo,center+speed
    pause 20
  next

  goto sense
```

program generates a random number and then executes this step:

```
for i=1 to delay +
(rnd&3*5)  ‘ turn a random
amount
```

This causes the entire loop to execute at least *delay* cycles. It also add a random element from 0 to 15 (3 is the maximum random number, and 3*5 is 15).

TROUBLESHOOTING

If your robot doesn’t work correctly, here are a few things to check.

The problem you are most likely to have with this project is correctly positioning the LEDs and the sensors. Be sure to position the sensors far apart and shield the LEDs completely using heat shrink tubing or a soda straw. The shield should completely block the sides of the LEDs only allowing light to escape in the forward direction. Of course, the sensors and LEDs should be as close to the front of the robot as possible.

If the robot’s motion seems jerky, be sure the 3300μF capacitor is in place. Without this capacitor, noise spikes from the motors can reset the BASIC Stamp, making the motion seem to start and stop. If you suspect this is the case, try placing the following line at the start of either Listing 6 or Listing 7:

```
   DEBUG “RESET”,CR
```

Then, when the robot is connected to your PC, you should only see this line execute once. If you see it multiple times, the BASIC Stamp is resetting spuriously.

If the sensors don’t seem to register objects, make sure they are wired correctly along with the LEDs. You might consider replacing the LEDs with a visible unit so you can see if the circuit is working (or better still, count the frequency with the BASIC Stamp or an oscilloscope).

If the robot detects objects very far away, you probably need to experiment with the location and direction of the LEDs and sensors. You might also consider using larger resistors with the LEDs (perhaps 2 kΩ) to reduce their output.

THE NEXT STEP

What else can you do with your Boe Bot? Using the information in last month’s article, you could easily keep a log of objects detected and store it in EEPROM. You could use a switch to turn off the robot’s motors and dump the object log to the debug terminal. This information might be useful if you were programming the robot to run through a maze.

Speaking of mazes, you could add more IR sensors (and LEDs) to allow the robot to sense objects along its sides as well as straight ahead. Using this extra information could allow the robot to solve the maze more quickly — an important consideration in competition with other robots.

(990050-4)

Internet

http://www.parallaxinc.com – BASIC Stamp Manual Version 1.9, BASIC Stamp DOS and Windows Editor, program examples. International distribution sources.

*http://www.stampsinclass.com —BoE documentation, Robotics curriculum, BoE-Bot *.dxf and *.dwg drawing formats, discussion group for educational uses of BASIC Stamp.*

chucks@turbonet.com — creator of the BoE-Bot and author of this series. Technical assistance.

kgracey@parallaxinc.com — co-author of this article. Technical assistance and questions about the educational program.

http://www.milinst.demon.co.uk — UK distributor of Parallax BASIC Stamp.